

Git 常用命令速查表

mater :默认开发分支

origin :默认远程版本库

Head :默认开发分支

Head^ :Head的父提交

创建版本库

\$ git clone <url>

#克隆远程版本库

\$ git init

#初始化本地版本库

分支与标签

\$ git branch

#显示所有本地分支

\$ git checkout <branch/tag>

#切换到指定分支或标签

\$ git branch <new-branch>

#创建新分支

\$ git branch -d <branch>

#删除本地分支

\$ git tag

#列出所有本地标签

\$ git tag <tagname>

#基于最新提交创建标签

\$ git tag -d <tagname>

#删除标签

修改和提交

\$ git status

#查看状态

\$ git diff

#查看变更内容

\$ git add .

#跟踪所有改动过的文件

\$ git add <file>

#跟踪指定的文件

\$ git mv <old> <new>

#文件改名

\$ git rm <file>

#删除文件

\$ git rm -cached <file>

#停止跟踪文件但不删除

\$ git commit -m "commit message"

#提交所有更新过的文件

\$ git commit -amend

#修改最后一次提交

查看提交历史

\$ git log

#查看提交历史

\$ git log -p <file>

#查看指定文件的提交历史

\$ git blame <file>

#以列表方式查看指定文件的提交历史

撤销

\$ git reset --hard HEAD

#撤销工作目录中所有未提交文件的修改内容

\$ git checkout HEAD <file>

#撤销指定的未提交文件的修改内容

\$ git revert <commit>

#撤销指定的提交

合并与衍合

\$git merge <branch>

#合并指定分支到当前分支

\$git rebase <branch>

#衍合指定分支到当前分支

远程操作

\$ git remote -v

#查看远程版本库信息

\$ git remote show <remote>

#查看指定远程版本库信息

\$ git remote add <remote> <url>

#添加远程版本库

\$ git fetch <remote>

#从远程库获取代码

\$ git pull <remote> <branch>

#下载代码及快速合并

\$ git push <remote> <branch>

#上传代码及快速合并

\$ git push <remote> :<branch/tag-name>

#删除远程分支或标签

\$ git push -tags

#上传所有标签

Git 的工作就是创建和保存你项目的快照及与之后的快照进行对比。

Git 常用的是以下 6 个命令：

git clone、git push、git add 、git commit、git checkout、git pull

The diagram illustrates the Git workflow using four colored cylinders: a green cylinder for 'workspace', a yellow cylinder for 'staging area', a blue cylinder for 'local repository', and an orange cylinder for 'remote repository'. Arrows indicate the following operations: 'add' from workspace to staging area; 'commit' from staging area to local repository; 'push' from local repository to remote repository; 'fetch/clone' from remote repository to local repository; 'pull' from remote repository to workspace; and 'checkout' from local repository to workspace.

warkspace: 工作区

staging area: 暂存区/缓存区

local repository: 版本库，或本地仓库

remote repository: 远程仓库

1. 创建仓库命令

git init 初始化仓库

git init命令用于在目录中创建新的git仓库，所有有关你的此项目的快照数据都存放在这里。

如：

例如我们在当前目录下创建一个名为 runoob 的项目：

```
-- mkdir product
```

```
-- cd product/
```

```
-- git init
```

– Initialized empty Git repository in /Users/tianqixin/www/runoob/.git**/**

初始化空 Git 仓库完毕。

现在你可以看到在你的项目中生成了 .git 这个子目录，这就是你的 Git 仓库了，所有有关你的此项目的快照数据都存放在这里。

.git 默认是隐藏的，可以用 ls -a 命令查看：

git clone 拷贝一个git仓库到本地，让自己能够查看该项目，或者进行修改。

git clone [url]

git会按照提供的url所指的项目的名称创建你的本地项目(通常是url最后一个/之后的项目名称)

git clone [url] authorName 按指定名称创建本地项目

2. 提交与修改

git add 将文件添加到暂存区

git add [file1] [file2] ... 添加一个或多个文件到缓存区

git add [dir] 添加指定目录到暂存区，包括子目录

git add . 添加当前目录下所有改动的文件到暂存区

git status -s

?? file1

?? file2

??表示该文件未添加到暂存区

执行完git add file1 file2

再执行

git status -s

A file1

A file2

A表示这两个文件已添加到暂存区

AM file1

A file2

AM的状态是指该文件添加到缓存区后又有改动

git status 查看上次提交之后是否对文件再次修改，查看状态

git status -s 普遍使用-s参数来获得简短的输出结果

git diff 比较文件的不同，即比较文件在暂存区和工作区的差异

显示已写入暂存区的和已经被修改但尚未写入暂存区文件的区别

git diff 尚未缓存的改动

git diff --cached 查看已缓存的改动

git diff HEAD 查看已缓存的与未缓存的所有改动

git diff --stat 显示摘要而非整个diff

显示暂存区和工作区的差异: git diff [file]

显示缓存区和上一次提交 (commit) 的差异 git diff --cache [file] 或 git diff --staged [file]

显示两次提交之间的差异 git diff [first-branch] ... [second-branch]

git commit命令 将暂存区的内容添加到本地仓库中

git commit

git commit -m '备注'

git commit [file1] [file2] ... -m [message] 将暂存区的指定文件提交到仓库区

git commit -a 设置修改后的文件不需要执行git add命令, 直接来提交

git commit -am [message]

git config 设置提交代码时的用户信息:

git config --global user.name 'lmz'

git config --global user.email test@xxx.com

如果去掉-- global参数, 则只对当前仓库有效

git reset 回退版本, 可以指定回退某一次提交的版本

git reset [--soft --mixed | --hard] [HEAD]

--mixed为默认, 可以不带此参数。用于充值暂存区的文件与上一次的提交(commit)保持一致

如: git reset [HEAD]

– git reset HAED 回退所有内容到上一个版本

– git reset HEAD file1 回退file1文件的版本到上一个版本

– git reset 052e 回退到指定版本

–soft参数用于回退到某个版本

git reset --soft HEAD

实例: git reset --soft HEAD~3 回退到上上上个版本

-hard参数 撤销工作区所有未提交的修改，将暂存区与工作区都回到上一个版本，并删除之前的所有信息提交

git reset --hard HEAD

git reset --hard HEAD~3 回退到上上个版本

git reset --hard bae128 回退到某个版本回退点之前的所有信息

git reset --hard origin/master 将本地的状态回退到和远程的一样

注意：谨慎使用 --hard参数，他会删除回退点之前的所有信息

HEAD 说明：

HEAD 表示当前版本

HEAD^ 上一个版本

HEAD^^ 上上一个版本

HEAD^^^ 上上上一个版本

可以使用 ~ 数字表示

HEAD~0 表示当前版本

HEAD~1 上一个版本

HEAD^2 上上一个版本

HEAD^3 上上上一个版本

以此类推...

git rm 删除工作区文件夹

3. 提交日志

git log 查看历史提交记录

在使用 Git 提交了若干更新之后，又或者克隆了某个项目，想回顾下提交历史，我们可以使用 git log 命令查看。

git log

可以用 --oneline 选项来查看历史记录的简洁的版本。

git log --oneline

我们还可以用 --graph 选项，查看历史中什么时候出现了分支、合并。以下为相同的命令，开启了拓扑图选项。

可以用 -reverse 参数来逆向显示所有日志。

查找指定用户的提交日志可以使用命令：git log --author

git log --author --reverse --online -5

git blame 以列表形式查看指定文件的历史修改记录

查看指定文件的修改记录使用 git blame 命令

git blame <file> git lame file1

4. 远程操作

git remote 增删改查远程仓库

git remote -v 显示所有远程仓库

git remote show [remote] 显示某个远程仓库的信息

如: git remote show <https://github.com/tianqixin/runoob-git-test>

git remote add [shortname] [url] 添加远程仓库

shortname为本地的版本库

如: 提交到 Github

git remote add origin git@github.com:tianqixin/runoob-git-test.git

git remote rm name 删除远程仓库

git remote rename old_name new_name 修改仓库

git fetch 从远程获取代码库

该命令执行完后需要执行 git merge 远程分支到你所在的分支。

git merge 从远端仓库体去数据并尝试合并到当前分支

该命令就是在执行 git fetch 之后紧接着执行 git merge 远程分支到你所在的任意分支。

git fetch [alias]

git merge [alias]/[branch]

然后我们在本地更新修改。

比如远端修改了某一文件, 然后本地进行git fetch

\$ git fetch origin

remote: Counting objects: 3, done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

From github.com:tianqixin/runoob-git-test

0205aab..febd8ed master -> origin/master

以上信息"0205aab...febd8ed master -> origin/master" 说明 master 分支已被更新, 我们可以使用以下命令将更新同步到本地:

\$ git merge origin/master

Updating 0205aab..febd8ed

Fast-forward

README.md | 1 +

1 file changed, 1 insertion(+)

git pull命令

从远程获取代码并合并本地的版本。

git pull 其实就是 git fetch 和 git merge FETCH_HEAD 的简写。命令格式如下：

git pull <远程主机名> <远程分支名>:<本地分支名>

实例

更新操作：

\$ git pull

\$ git pull origin

将远程主机 origin 的 master 分支拉取过来，与本地的 brantest 分支合并。

git pull origin master:brantest

如果远程分支是与当前分支合并，则冒号后面的部分可以省略。

git pull origin master

上面命令表示，取回 origin/master 分支，再与本地的 brantest 分支合并。

上面的 pull 操作用 fetch 表示为：

以我的 <https://github.com/tianqixin/runoob-git-test> 为例，远程载入合并本地分支。

\$ git remote -v # 查看信息

origin https://github.com/tianqixin/runoob-git-test (fetch)

origin https://github.com/tianqixin/runoob-git-test (push)

\$ git pull origin master

From https://github.com/tianqixin/runoob-git-test

*** branch master -> FETCH_HEAD**

Already up to date.

上面命令表示，取回 origin/master 分支，再与本地的 master 分支合并。

git push 将本地的分支版本上传到远程并合并

命令格式如下：

git push <远程主机名> <本地分支名>:<远程分支名>

如果本地分支名与远程分支名相同，则可以省略冒号：

git push <远程主机名> <本地分支名>

实例

以下命令将本地的 master 分支推送到 origin 主机的 master 分支。

\$ git push origin master

相等于：

\$ git push origin master:master

如果本地版本与远程版本有差异，但又要强制推送可以使用 --force 参数：

git push --force origin master

删除主机的分支可以使用 --delete 参数，以下命令表示删除 origin 主机的 master 分支：

git push origin --delete master

以我的 <https://github.com/tianqixin/runoob-git-test> 为例，本地添加文件：

\$ touch runoob-test.txt # 添加文件

\$ git add runoob-test.txt

\$ git commit -m "添加到远程"

master 69e702d] 添加到远程

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 runoob-test.txt

\$ git push origin master # 推送到 Github

将本地的 master 分支推送到 origin 主机的 master 分支。

版权声明：本文为CSDN博主「~」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin_45215832/article/details/123564927